

TEMA 5: PROTOCOLOS DE COMUNICACION.

1. MECANISMOS DE CONTROL DE LA COMUNICACIÓN.

Además de estar pensando para el nivel de enlace (nivel 2) también se puede aplicar el nivel de transporte (nivel 4). El nivel de enlace da una comunicación fiable entre máquinas adyacentes conectadas mediante una línea física. Sobre este protocolo se apoyan los niveles superiores. El nivel otorga eficiencia (control de flujo) y fiabilidad.

• TIPOS DE CONFIGURACION DEL NIVEL DE ENLACE:

-PUNTO A PUNTO: Se conectan solo dos ordenadores a la misma línea física (no hace falta direccionar).



-MULTIPUNTO: N máquinas se conectan al mismo medio físico. Se necesita direccionar para indicar quien es el destinatario de la trama.

La configuración más habitual es Maestro- Esclavo (Primaria-Secundaria)

* **MAESTRO:** Regula el uso de la línea. Decide los turnos de dialogo para que no haya más de una línea transmitiendo a la vez.

* **ESCLAVO:** Solo puede transmitir un esclavo a la vez, pero si pueden transmitir un esclavo y el maestro a la vez.



Debido a que la información que se envía por el medio, se puede escuchar por todos los equipos conectados al medio, hay varios servicios para controlar el flujo de datos y los errores.

-ORIENTADO A CONEXION: Suele ser confirmado, es decir, consta de conexión, envío y liberación.

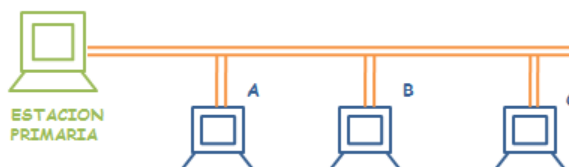
-NO ORIENTADO A CONEXION: No confirmado, solo se envía información.

1.1 COORDINACION DE LA COMUNICACIÓN.

En un enlace multipunto, tenemos que establecer cómo y quién va a transmitir en cada momento.

• **CENTRALIZADA:** Tenemos una estación primaria que se encarga de coordinar la transmisión y varias estaciones secundarias.

Se basa en dos operaciones o fases de comunicación: SONDEO/SELECCION.



-SONDEO: La estación primaria comprueba si alguna de las estaciones secundarias quiere solicitar un envío de información.

La estación primaria es la única que puede transmitir. Para que una estación secundaria pueda transmitir debe ser sondeada, es decir, la estación primaria pregunta una a una a cada estación secundaria si quiere establecer comunicación.

Aunque una estación quiera transmitir, hasta que no se la sondee no podrá.

En una transmisión los datos se envían a la primaria independientemente de que sea para ella o no y después la estación primaria los reenvía.

-SELECCION: La estación primaria quiere enviar algo a una estación secundaria.

Transmite información a una estación secundaria en concreto, el resto de estaciones escuchara la transmisión pero no hará nada.

Reenvía datos recibidos en la fase de sondeo a la estación destino real de la información.

- **DISTRIBUIDA:** Son estaciones balanceadas, usa filosofía Ethernet.

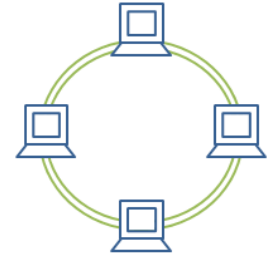


Lo que se envía por el medio, lo escuchan todos los equipos conectados al medio y deciden si es para ellos.

Se pueden producir colisiones.

- **TOKEN-RING:** Manda a la primera estación la información, sino es para ella la envía a la siguiente. Si la información es para la estación se la queda.

No se producen colisiones y las estaciones no comparten el medio de transmisión.



1.2 DELIMITACION DE TRAMAS.

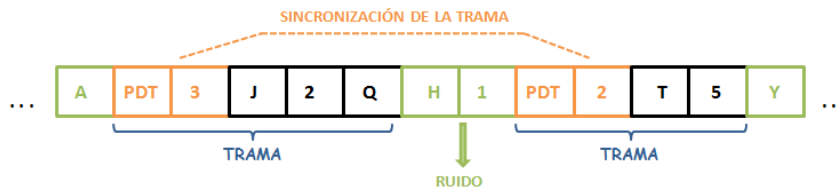
Las entradas de nivel de enlace dividen el conjunto de bits a transmitir por el nivel físico en tramas. Hay dos tipos de tramas: información y control.

Los métodos de realizar la sincronización de la trama y delimitarlas son:

- **PRINCIPIO Y CUENTA:** Establece el principio y se cuenta los caracteres que se envían.

-**PDT:** Carácter de comienzo de trama.

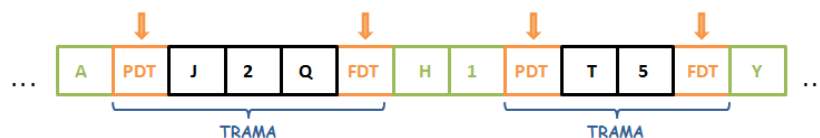
Entre trama y trama se cuelan caracteres que no tienen sentido, se les consideran RUIDO y se van a omitir,



-**TRANSPARENCIA:** No es necesaria, ya que la cuenta de caracteres marca la trama y PDT puede enviarse dentro de la trama ya que los caracteres de la trama no se interpretan.

- **INDICADOR COMIENZO Y FIN:** Establece el principio y el fin de la trama, mediante una secuencia de caracteres (PDT y FDT)

Se usa en protocolos orientados a carácter.



-**TRANSPARENCIA:** Necesaria, para poder distinguir entre un FDT que va dentro de una trama como información y el indica el fin de la trama.

- **GUIONES:** Se usa un conjunto de bits específico para indicar el principio y el final de una trama.

Se usa en protocolos a nivel de bit.

-**GUION INICIO = GUION FINAL:** 0 1 1 1 1 1 0



1.3 TRANSPARENCIA.

Mecanismo que se implementa en un protocolo que garantiza que cualquier dato se pueda enviar por una trama, incluso, delimitadores.

Las técnicas que garantizan la transparencia son:

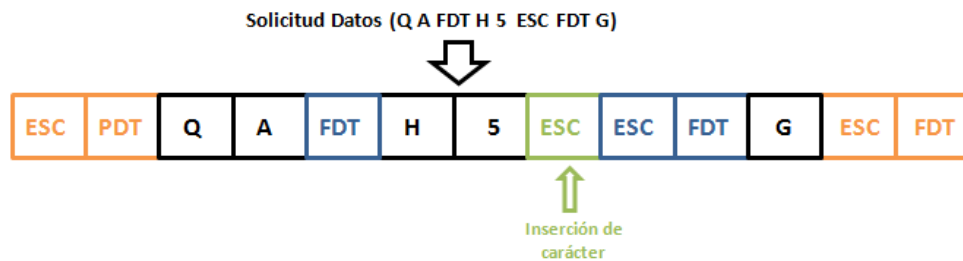
- **INSERCCION DE CARACTER:**

Delante del delimitador se añade un ESC para indicar que si detrás hay un delimitador, el delimitador está indicando fin de trama.

Cuando se encuentra en ESC en medio de la trama se va a insertar otro ESC.

Delante de un carácter de control solo, no se hace nada porque no significa nada, es solo un carácter de información que se envía en la trama.

-**AL INTERPRETAR:** Cuando se encuentran dos ESC seguidos, se elimina uno de los ESC y lo interpreta como un carácter dentro de la trama.

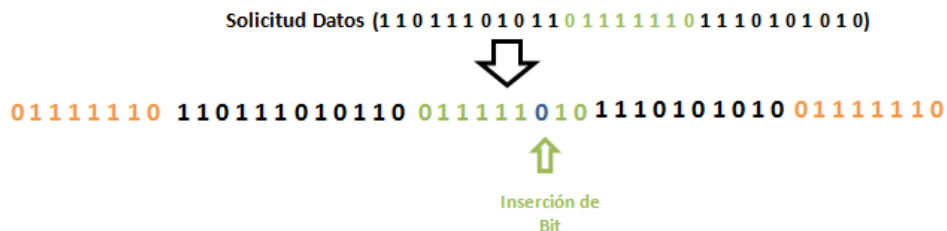


- **INSERCCION DE BIT:**

-**ORIGEN:** Mira lo que quiere enviar. Cuando encuentra el guion dentro de la trama se va a insertar un cero antes del último uno, es decir, se va a insertar un cero cada vez que se localicen cinco unos seguidos.

Cuando el origen encuentra un cero y 5 unos añade un cero para hacer la transparencia.

-**DESTINO:** Elimina el cero cuando se encuentran 5 unos seguidos, un cero y otro uno.



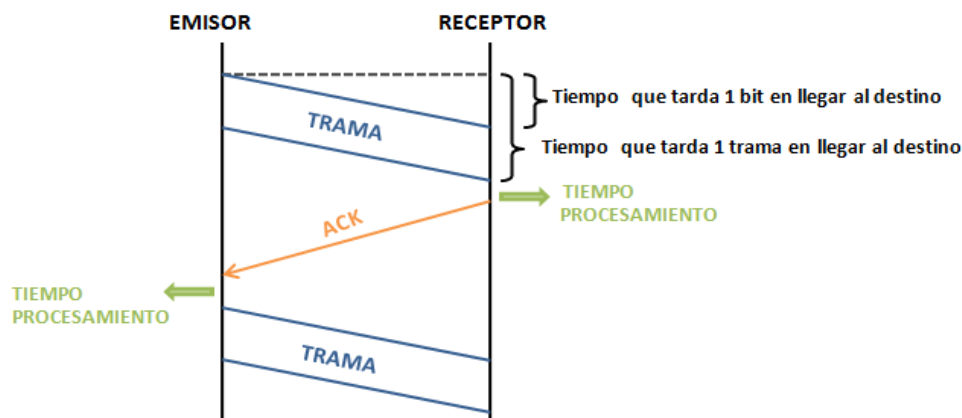
La transparencia se hace sobre los datos a enviar, los guiones se añaden después de hacerla.

1.4 CONTROL DE FLUJO.

Función por la que se asegura que una estación emisora no inunde con datos a una estación receptora.

El destino debe indicar al origen cuando no envía más información porque está saturada y luego restablecer la comunicación.

- **PARADA Y ESPERA:** Técnica fácil y sencilla de implementar.



-FUNCIONAMIENTO:

1. Emisor transmite una trama
2. Receptor después de que la trama llegue, envía otra trama para indicar al emisor que puede continuar.

Se pierde mucho tiempo, ya que no se puede enviar ninguna trama más hasta que no llegue la confirmación del destino.

Si el tiempo de transferencia de bit es mínimo, sólo perderíamos el tiempo de enviar una trama.

Si la trama es enorme, entonces no se perdería casi tiempo porque las tramas se envían casi continuamente.

• VENTANA DESLIZANTE:

Emisor podrá enviar N tramas (antes del tiempo de procesamiento de la primera trama) sin necesidad de confirmarlas.

$\text{NUMERO_TRAMA}(N) = \text{TAMAÑO DE LA VENTANA DE TRANSMISION.}$

Emisor reserva memoria para almacenar las tramas, por si no han llegado bien al destino.

Se usa un número de secuencia en la trama para evitar errores cuando se pierde alguna confirmación ya que si no podrían producirse confusiones.

Las confirmaciones del receptor rotan la ventana del emisor autorizando más transmisiones.

El receptor usa dos tipos de tramas:

- * **RR**: Listo para recibir más tramas.
- * **RNR**: El receptor se ha saturado y todavía no está lista para recibir más.

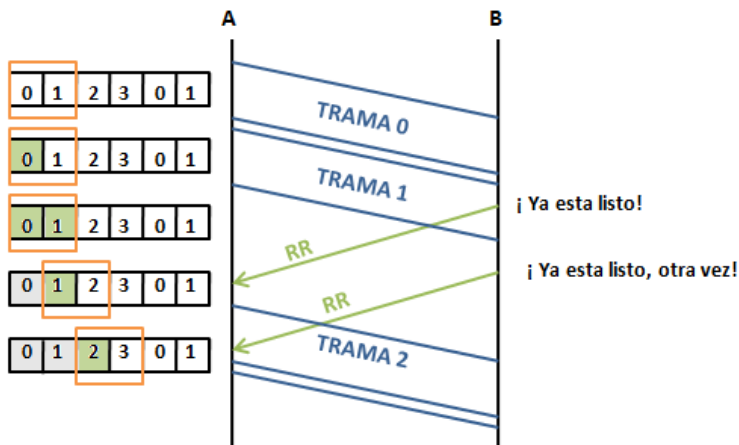
-VENTANA:

El tamaño de ventana es el número máximo de tramas que se envían a la vez.

Se comienza la numeración de tramas por 0, por tanto el número de tramas va desde 0. N-1.

A mayor tamaño de ventana, mayor aprovechamiento.

Cuando se aprovecha todo el medio de transmisión se dice que el envío es continuo (aprovechamiento optimo).

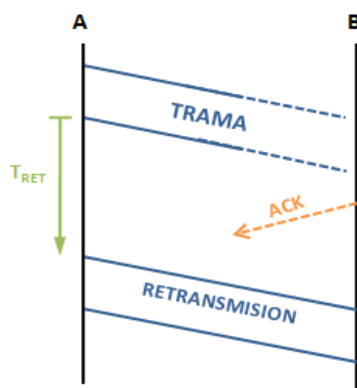


• RECUPERACION DE ANOMALIAS:

-**PERDIDA DE TRAMA**: Cuando se pierde una trama, el receptor no va a saber ni siquiera que se ha enviado algo.

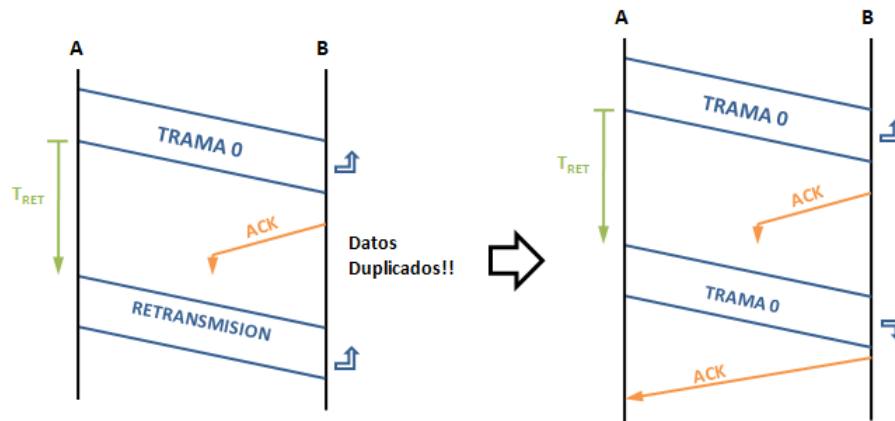
No se envía ninguna confirmación, por tanto, se va a necesitar un temporizador de retransmisión (T_{RET}).

Cuando vence el temporizador entonces se retransmitirá la trama.



-PERDIDA DE CONFIRMACION: Como no llega el ACK, cuando vence el temporizador se va a reenviar la trama.

El receptor va a creer que los datos son nuevos y va a duplicar la información.

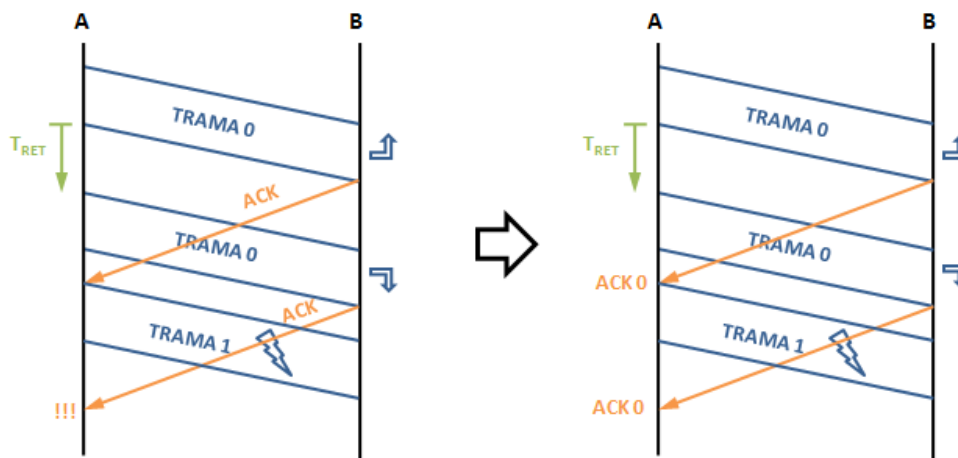


El emisor solo espera una confirmación, pero no sabe lo que ha pasado durante el envío. La solución es numerar las tramas, así cuando el receptor recibe una trama duplicada la deshecha y vuelve a mandar el ACK.

-VENCIMIENTO PREMATURO DE TEMPORIZADOR: El temporizador de retransmisión vence antes de que llegue la confirmación desde el emisor.

El primer ACK, hace que se envíe la trama 1. Si suponemos que la trama 1 se pierde cuando llegue el ACK de la retransmisión se va a interpretar como que la trama 1 ha llegado bien.

La solución será numerar los ACK, de esta manera la numeración nos va a permitir saber exactamente cuál es la trama que ha llegado correctamente.



La solución será numerar los ACK, de esta manera la numeración nos va a permitir saber exactamente cuál es la trama que ha llegado correctamente

1.5 CONTROL DE ERRORES.

Para realizar el control de errores, el destino detecta el error y solicita retransmisión, hay varios métodos:

- **FEC:** Destino detecta y corrige el error sin necesitar retransmisiones.

Muy complejo

Se añade redundancia al mensaje, es decir, se añaden bits adicionales que permiten detectar el error y corregirlo.

- **DETECCION DE ERRORES:**

-COMPROBACION DE PARIDAD: Se usa en ASCII y consiste en añadir un bit de paridad al final del bloque de datos.

* **PARIDAD IMPAR:** El valor del bit añadido se determina de modo que el número total de 1's sea impar.

* **PARIDAD PAR:** El valor del bit añadido se determina de modo que el número total de 1's sea par.

Detecta un número impar de errores pero no los corrige.

En ASCII para codificar un carácter se usan 7 bits más 1 bit de paridad.

-**CRC**: Dado un mensaje de m bits, el emisor genera una secuencia de r bits (SVT) tal que $m \gg r$.

La trama resultante ($m+r$ bits) será divisible por algún número determinado.

El receptor va a dividir la trama por el número, si no es exacto entonces se habrá producido un error.

* **CODIGOS POLINOMICOS**: Representa las ristas de bits como polinomios con coeficientes binarios.

Sea $M(x)$ = mensaje original (m bits)

$G(x)$ = polinomios generador ($r+1$ bits)

$T(x)$ = Mensaje a transmitir ($m+r$ bits)

En la emisión: $T(x) = M(x) \cdot x^r \text{ XOR } R(x)$ donde $R(x) = \text{MOD}(M(x) \cdot x^r / G(x))$

En la recepción: $R'(x) = \text{MOD}(T(x)/G(x)) \Rightarrow \begin{cases} R'(x)=0 \Rightarrow \text{No error} \\ R'(x) \neq 0 \Rightarrow \text{Error} \end{cases}$

* **ERRORES DETECTADOS**:

Errores de 1 bit

Errores dobles $\Leftrightarrow G(x)$ al menos tiene 3 1's.

Número impar de errores $\Leftrightarrow G(x)$ con factor $(x+1)$

Ráfagas de errores de longitud menor que $G(x)$ y la mayoría de longitud mayor.

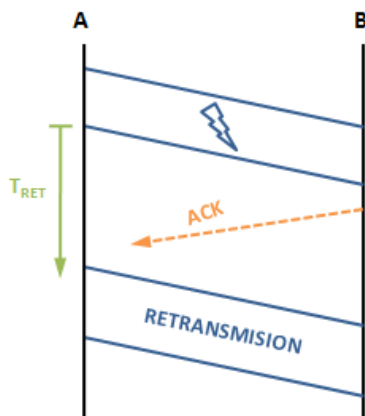
-**ARQ**: Destino detecta el error y solicita al origen que retransmita la trama.

• **ARQ CON PARADA Y ESPERA**:

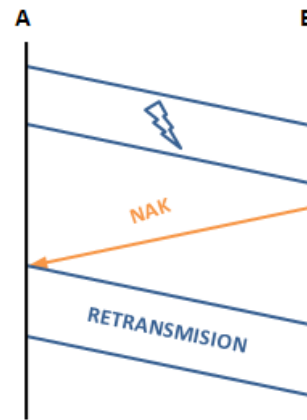
El destino no hace nada, solo envía confirmaciones, el ACK nunca se manda para la retransmisión sino que tiene que vencer el temporizador.

Si además queremos que envíe rechazos, cuando el mensaje llega con errores el destino mandará un NAK para la retransmisión.

CONFIRMACIONES



CONFIRMACION Y RECHAZO



• **ARQ CON VENTANA DESLIZANTE Y RECHAZO SIMPLE**:

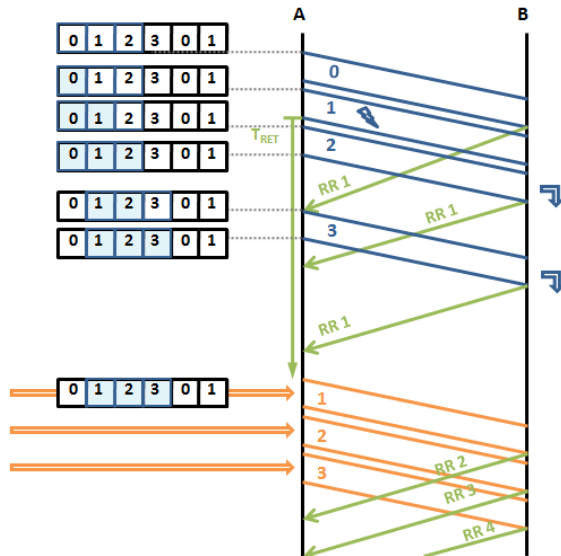
-**TEMPORIZADOR**: Retransmite por vencimiento del temporizador.

Cuando una trama llega bien, el destino envía un RR con la secuencia de la trama esperada.

Cuando una trama se pierde o llega con errores el destino no hace nada. Si después llega bien otra trama el destino va a seguir esperando la trama errónea.

A partir de que una trama llega con errores, el resto de las tramas que van llegando a continuación se descartan.

Con cada nuevo RR se va a rotar una posición la ventana.



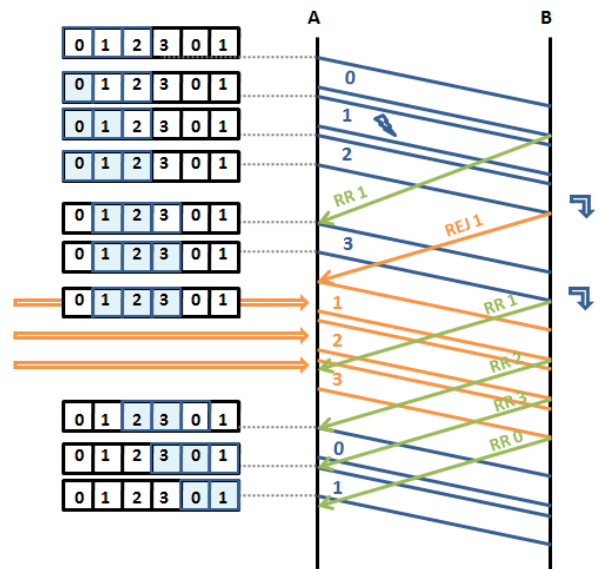
Cuando vence el temporizador, se va a mirar el último RR y se van a retransmitir toda la ventana desde ese punto.

-TRAMAS DE RECHAZO: Usa tramas de rechazo, en vez de esperar a que venza el temporizador.

Cuando el destino detecta una trama fuera de secuencia, se envía una trama REJ y esta activa la retransmisión desde la trama perdida.

La trama REJ provoca un reenvío, no rota la ventana.

A partir de que llega una trama errónea, el resto de tramas se desechan.



Cuando llega un RR con un numero de secuencia x, se va a rotar la ventana x veces. Los RR valida el envío de la trama a la que representa y todas las anteriores sin importar si algún RR se había perdido.

-TAMAÑO MAXIMO VENTANA DE TRANSMISION (WT):

$$\text{MAX WT} = 2^n - 1 \implies \text{Nº_Sec} = 0 \dots 2^n - 1 \text{ donde: } n \text{ bits necesarios para codificar}$$

Si el tamaño de la ventana es igual al número de secuencia, cuando enviemos la ventana entera si se pierden todos los RR no se va a poder detectar.

La ventana de recepción en esta caso es siempre 1 (WR=1)

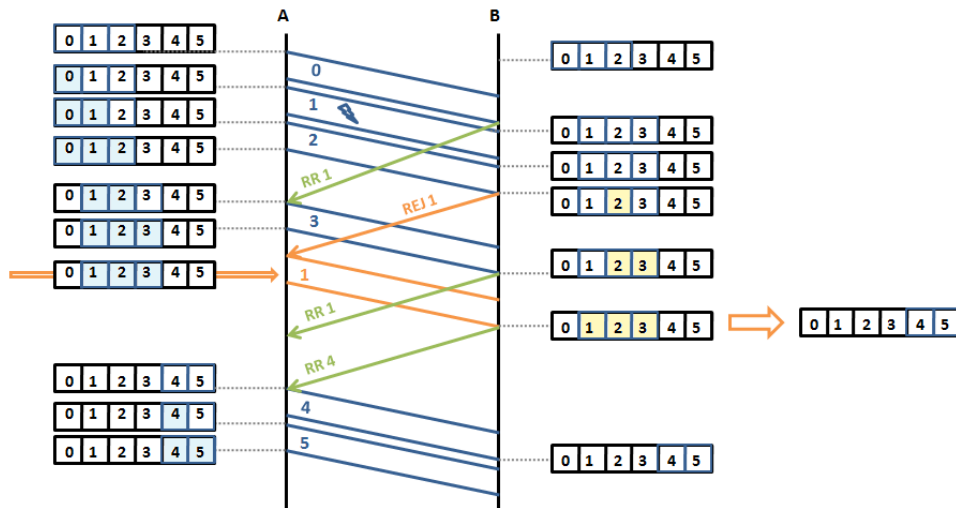
• **ARQ CON RECHAZO SELECTIVO:** Solo provoca la retransmisión de la trama errónea.

Se necesita ventana de recepción para que se puedan almacenar las tramas que se envían.

Si la ventana no está llena (hay algún hueco por que se ha perdido alguna trama) no se rotara la ventana.

Cuando se pierde una trama se va a enviar un SREJ que hace que se reenvíe otra vez la trama perdida.

Los RR que envía el destino van a indicar el extremo inferior de la ventana de recepción.

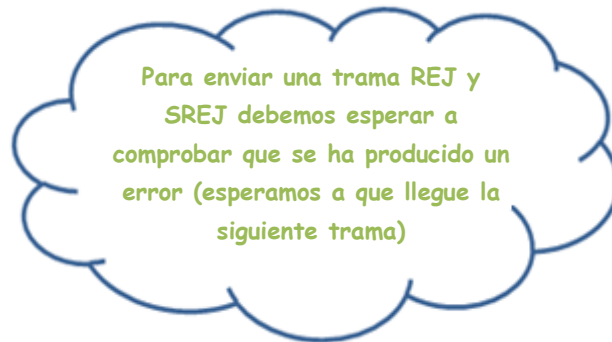


Cuando llega la trama, se rellena el hueco en la ventana de recepción (WR) con los RR que envía el destino.

-TAMAÑO MAXIMO VENTANA DE RECEPCION (WR):

$$\text{MAX WR} = \text{MAX WT} = 2^n/2 \quad \Rightarrow \quad \text{N}^\circ_Sec = 0 \dots 2^n/2 \quad \text{donde: } n \text{ bits necesarios para codificar}$$

Cuando rota la ventana, la nueva ventana no tiene que tener ningún número de secuencia igual que en la ventana anterior, ya que si no se producirían problemas cuando se pierden los RR.

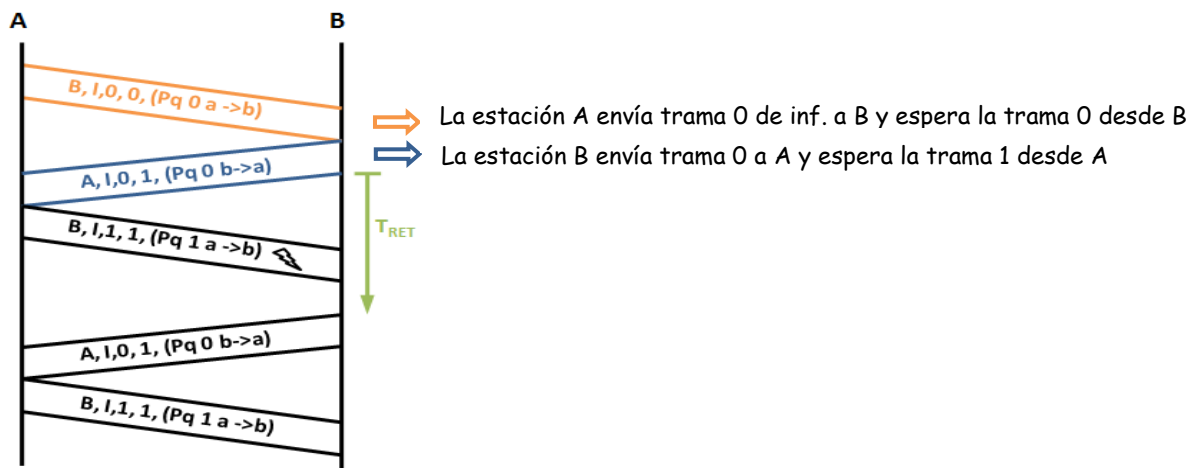


1.6 TRANSMISION BIDIRECCIONAL DE DATOS.

La transmisión bidireccional de datos se puede hacer introduciendo un campo de clase de trama que indique si la trama es de información, de confirmación. También se puede usar la técnica del piggy_backing.

- **PIGGY_BACKING:** Se pueden enviar en la trama de información hacia el destino y confirmar al origen a la vez.

Aprovecha las tramas de datos enviadas para incorporar las confirmaciones de las tramas de datos recibidas.



Cuando se usa piggy_backing las tramas de confirmación solo se van a utilizar cuando no hayan datos que enviar a la otra estación, siempre que queden datos por enviar, las confirmaciones irán en la propia trama de información.

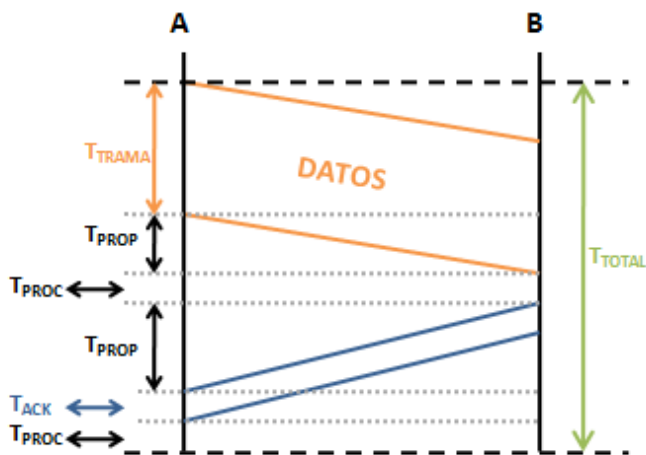
2. EFICIENCIA DE UN PROTOCOLO.

Mide el grado de aprovechamiento de la capacidad del canal de comunicación necesita ventana. El grado de aprovechamiento se mide a través de la siguiente formula.

* **GRADO APROVECHAMIENTO** $\Leftrightarrow U = \frac{T_{\text{Aprovechado}}}{T_{\text{total}}}$

* **TIEMPO TRANSMISION** $\Leftrightarrow T_{\text{trans}} = \frac{T_{\text{ideal}}}{U}$ donde: $T_{\text{ideal}} = \frac{\text{Tamaño_Transmitir}}{\text{Capacidad}}$

El tiempo que se está inactivo y se podría estar enviando información provoca ineficiencia en el protocolo.



* $T_{\text{total}} = 2T_{\text{prop}} + 2T_{\text{proc}} + T_{\text{trama}} + T_{\text{ack}}$

$$\text{Num_Tramas} = \frac{\text{Datos_Transmitir}}{\text{Tamaño_Trama}}$$

- **TIEMPO_ACK:** Tiempo de transmisión de la trama de confirmación, por muy pequeño que sea tarda un tiempo (tiempo en transmitir un bit).

* $T_{\text{ack}} = \text{Tamaño_ack} / \text{Capacidad (Vel_trans)}$.

El tamaño de la trama ack, va a ser el numero de bit que ocupa una trama de control.

- **TIEMPO_PROCESO:** Desde que recibes el último bit de la trama hasta que envías el primer bit de la confirmación.
- **TIEMPO_PROPAGACION:** Tiempo que transcurre desde que se envía el ultimo bit desde la estación emisora hasta que se recibe en la estación receptora.

* $T_{\text{prop}} = \text{Distancia} \times \text{Velocidad_Prop}$

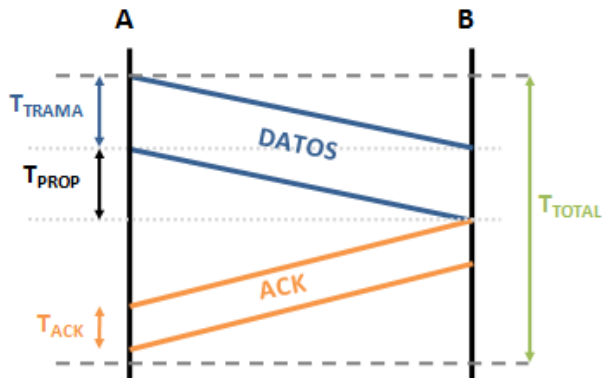
- **TIEMPO_TRAMA:** Tiempo que transcurre desde que se envía el primer bit de la trama hasta que se envía el ultimo bit.

* $T_{\text{trama}} = \text{Tamaño_trama} / \text{Capacidad (Vel_trans)}$ donde:

$T_{\text{am_trama}} = \text{Longitud_Datos} + \text{Longitud_cabecera}$

2.1 PARADA Y ESPERA.

Vamos a suponer que el tiempo de proceso y el tiempo de envío de la trama de confirmación son despreciables.



$$U = \frac{T_{\text{Aprovechado}}}{T_{\text{total}}} = \frac{T_{\text{Trama_util}}}{2T_{\text{prop}} + T_{\text{trama}}}$$

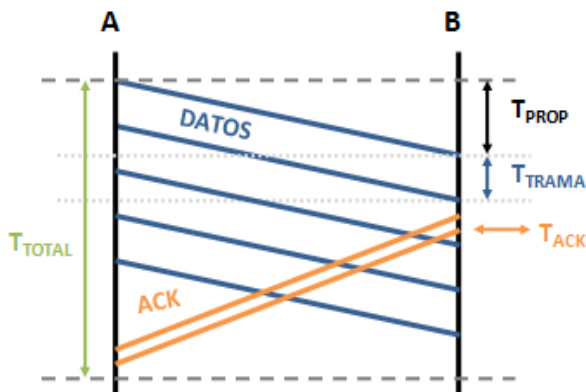


$$U = \frac{T_{\text{trama}}}{2T_{\text{prop}} + T_{\text{trama}}}$$

- **TIEMPO_TRAMA_UTIL:** Como el tiempo de envío del bit de control es despreciable, el tiempo de trama útil será el tiempo de envío de una trama de datos.

2.2 VENTANA DESLIZANTE.

- **SIN ENVIO CONTINUO:** Vamos a suponer que el tiempo de proceso y el tiempo de envío de la trama de confirmación son despreciables.

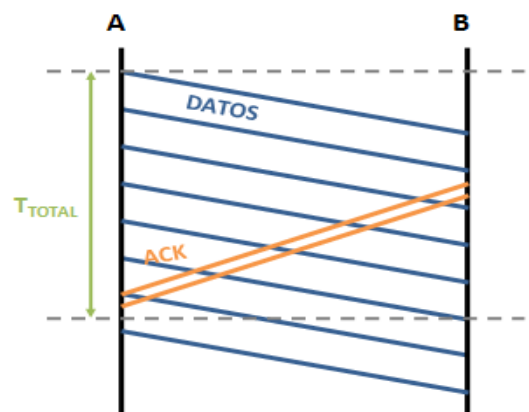


$$U = \frac{WT \times T_{\text{trama_util}}}{2T_{\text{prop}} + T_{\text{trama}}}$$

- **ENVIO CONTINUO:** En este caso $U \approx 1$, eso es debido todo el tiempo de la transmisión se aprovecha. Vamos a suponer que el tiempo de proceso y el tiempo de envío de la trama de confirmación son despreciables. El tamaño de ventana de transmisión va a ser máximo.

$$1 = \frac{WT \times T_{\text{trama_util}}}{T_{\text{total}}} \Leftrightarrow WT = \frac{T_{\text{total}}}{T_{\text{trama}}}$$

$$\Rightarrow WT = \frac{2T_{\text{prop}} + 2T_{\text{proc}} + T_{\text{trama}} + T_{\text{ack}}}{T_{\text{trama}}}$$



3. PROTOCOLOS DE NIVEL DE ENLACE.

3.1 PROTOCOLO LAPD.

Es el protocolo de nivel de enlace que se usa en el canal D de la red RDSI.

Trabaja en modo balanceado, con rechazo simple y las tramas I solo se usan como orden no como respuesta. (HDLC BA 2,8).

• CARACTERÍSTICAS:

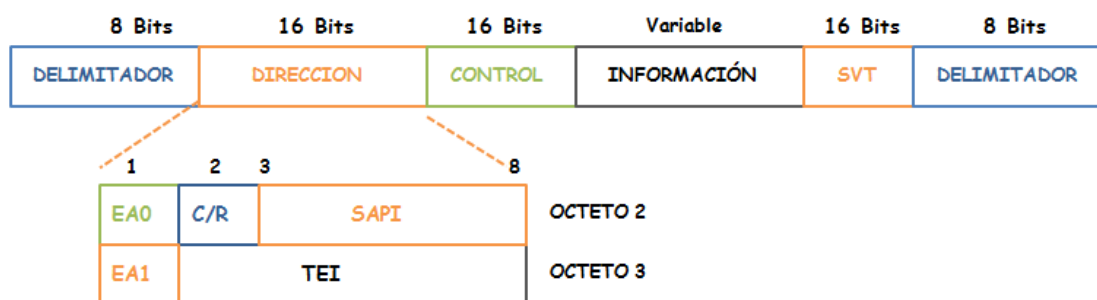
Modulo extendido

El campo de dirección es de 2 octetos, identifican el terminal y el SAP del nivel superior

Utiliza tramas XID Y UID.

Soporta varios enlaces lógicos: podemos multiplexar un nivel de enlace con varios niveles de red.

• FORMATO:



Este protocolo tiene dos octetos en el campo de dirección utilizados para funciones de multiplexación de enlaces lógicos.

El campo de dirección identifica un terminal específico (TEI) y un punto de acceso al servicio (SAPI). Se compone por los siguientes campos:

-EA0: Indica que el campo de dirección se extiende un octeto.

-EA1: Indica que el campo de dirección termina.

-SAPI: Identifica el SAP (protocolo) del nivel superior, es decir, del nivel 3.

Identifica un proceso distinto del nivel 3 (procesos de señalización, administración, gestión, datos de usuario modo paquete...).

-TEI: Identifica a cual de los dispositivos conectados a la RDSI va el paquete.

Representa un terminal específico o conjunto de terminales, por ejemplo, si tuviéramos 5 teléfonos y quisiéramos que los cinco sonasen a la vez, si se podría hacer ya que todos tienen el mismo TEI.

-C/R: Identifica si la trama que se envía es una orden o respuesta.

De esta manera no se tiene que mirar el campo de dirección para saber si es la información se envía desde la máquina origen o la destino.

Las tramas de información como respuesta se eliminan para que no se puede conocer el estado de la estación destino.

3.2 PROTOCOLO LAPF.

Es el protocolo que se utiliza como protocolo de nivel de enlace en frame_relay.

• CARACTERÍSTICAS:

El protocolo LAPF está a modo ABM.

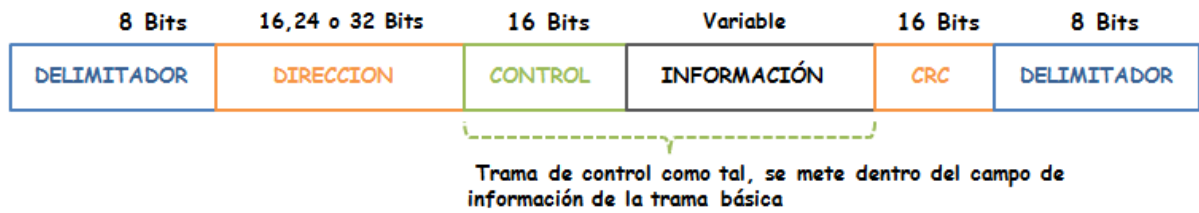
Utiliza modulo extendido.

La SVT (secuencia de verificación de trama) es un CRC de 16 bits.

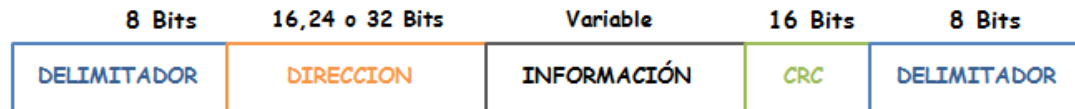
El campo de dirección tiene una longitud de 2, 3 o 4 octetos, que permite identificar la conexión del enlace de datos (DLCI) de 10, 16 o 23 bits.

- **PROTOCOLOS:**

-**PROTOCOLO CONTROL (TRAMA CONTROL):** Similar al protocolo HDLC.



-**PROTOCOLO BASICO (TRAMA BASICA):** Subconjunto del protocolo de control.



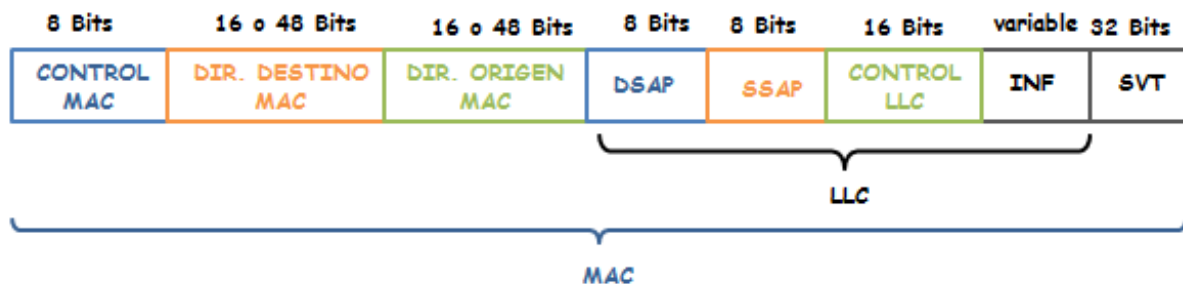
Se simplifica eliminando el campo de control, esto hace que no pueda asegurar ni la eficiencia ni la fiabilidad.

3.3 PROTOCOLO LLC.

Soluciona la comunicación con los protocolos de nivel3 en redes de área local ya que el nivel MAC no contempla a quien va dirigida la trama.

Este protocolo se encapsula dentro de la trama MAC.

- **FORMATO:**



* **DSAP:** SAP origen.

* **SSAP:** SAP destino.

- **SUBNIVELES DEL NIVEL DE ENLACE:**

-**SUBNIVEL MAC:** Medium Access Control

Incluye las direcciones origen y destino para identificar los dispositivos conectados a la red de área local.

Realiza la detección de errores usando código CRC de 32 bits.

Realiza funciones de control de acceso al medio mediante un campo de control.

-**SUBNIVEL LLC:** Logical Link Control

Incluye los puntos de acceso al servicio origen y del destino mediante los campos DSAP y SSAP que identifican al usuario lógico, es decir, el protocolo del nivel 3 de LLC en los sistemas origen y destino.

El campo de control tiene el mismo formato que HDLC, pero limitando los números de secuencia a 7 bits.

No realiza control de errores ni control de flujo.

3.4 PROTOCOLO LLC.

Se utiliza como protocolo de nivel de enlace en Internet.

Proporciona un método estándar para transportar datagramas multiprotocolo sobre enlaces punto a punto. Se usa en la conexión de usuarios a proveedores (líneas conmutadas) y entre encaminadores (líneas dedicadas).

• CARACTERÍSTICAS:

Detección de errores.

Reconoce múltiples protocolos y puede enviar información de nivel 2, tanto del protocolo IP como de otros protocolos.

Permite negociar la dirección IP en el momento de la conexión.

Verificación de autenticidad. Implementa un mecanismo de control de acceso.

• FORMATO DE TRAMA:



-**DIRECCION (FF)**: En una línea punto a punto no se necesita la dirección ya que solo ira la trama a un único equipo.

-**CONTROL (03)**: Al igual que en el caso de la dirección, en este protocolo no se necesita control.

-**PROTOCOLO**: Indica que es lo que transporta la trama.

* **DATAGRAMA IP**.

* **LCP**: Control de enlace

Se utiliza para negociar parámetros sobre el enlace: ventana de transmisión, temporizadores, opciones HDLC a usar ...

* **NCP**: Familia de protocolos de control de la red.

Ligado a los protocolos de nivel 3 que se estén utilizando, por ejemplo, si se usa IP negociara quien es el DNS, la IP, la máscara ...)

Tanto el campo dirección y el campo control pueden ser eliminados si previamente NCP lo negocia para no ser transmitidas. Es lo normal.

4. PROTOCOLOS DEL NIVEL DE TRANSPORTE.

4.1 PROTOCOLO TCP.

• CARACTERÍSTICAS:

1. Ofrece fiabilidad extremo a extremo.
2. Transferencia de flujo de octetos (byte-stream). TCP no recibe segmentos de datos si no un flujo continuo de bytes que va agrupando en segmentos para transmitir los datos.
TCP calcula un MSS (campo de datos del segmento) de tal forma que los datagramas IP se correspondan con la MTU de la red de acceso.
3. Ofrece un servicio orientado a conexión, por tanto realiza control de errores y control de flujo. Evita que un extremo no congestionen al otro, pero puede no hacer nada por no congestionar a los nodos intermedios, ya que los controles son extremo a extremo.
4. Multiplexado, es decir, puede dar servicio a varios procesos de aplicación simultáneamente, gracias a los números de puerto que los identifican.
5. Permite transferencias simultáneas en ambos sentidos, es decir, transferencia full-duplex.

• SEGMENTO TCP:



-**DATOS:** Incluye la cabecera de información del nivel de aplicación y los datos del mensaje (si existen).

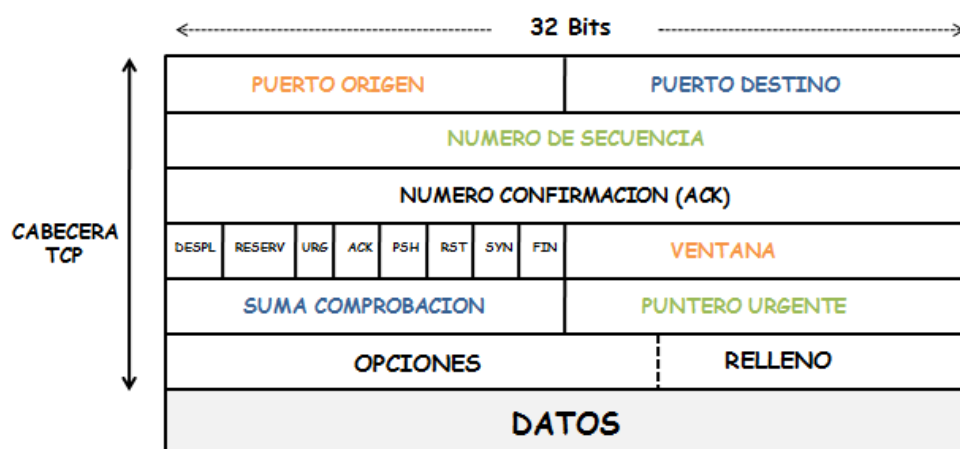
* **MSS:** Tamaño establecido con anterioridad a la transferencia que indica el máximo número de bytes que se pueden enviar en un segmento.

Se calcula de tal forma que los datagramas IP que se vayan a enviar coincidan con la MTU de la red.

* **DE OCTETOS A SEGMENTOS:**

1. Primero se guarda una copia de los datos que se quieren transmitir por si hay problemas en el envío y se necesita retransmitir.
2. Una vez que la entidad emisora ha llenado el buffer de transmisión, toma un trozo de los datos almacenados y les añade una cabecera para generar el segmento.

-**CABECERA:** Tiene longitud mínima, por omisión, de 20 octetos y máxima de 60 octetos incluyendo las opciones.



* **PUERTO ORIGEN/DESTINO:** Limitados a 16 bits cada uno. Como máximo 65536 puertos en un equipo. Identifican al proceso de aplicación emisor / receptor que envía un segmento TCP.

* **NUMERO DE SECUENCIA:** Indica el primer octeto del campo de datos del segmento que se envía. No se empieza a contar desde 0, sino desde un número aleatorio (por seguridad). Como el número de secuencia es de 32 bits, se va a trabajar en módulo 2^{32} , por tanto los números de secuencia se numeran cíclicamente del 0 al $2^{32}-1$.

* **NUMERO DE CONFIRMACION:** Indica el primer octeto del campo de datos del siguiente segmento que se espera recibir. Confirma todos los segmentos hasta el número de secuencia recibido.

* **DESP:** Indica el número de bloques de cuatro octetos que ocupa la cabecera. Como mínimo 20 octetos y como máximo 60 octetos.

* **RESERVADO:** Reservados para un uso futuro.

* **URG:** Si está activo indica que el campo de puntero urgente es un campo válido y significativo y que por tanto la entidad TCP receptora lo debe analizar debidamente.

* **PSH:** Servicio forzado de transferencia, es decir, se ha tenido que enviar el contenido del buffer de transferencia sin que se haya llenado y por tanto la entidad receptora tampoco esperará a que se llene el suyo.

* **RST:** Bit de reinicio. Si está activo indica al módulo TCP receptor que abandone la comunicación debido a un error o a una situación anormal. Se puede usar como respuesta para rechazar una solicitud de establecimiento de una conexión TCP.

* **FIN:** Si está activo indica que se está solicitando finalización o liberación del enlace.

* **ACK**: Indica que el campo de numero de confirmación es un campo significativo.

* **SYN**: Si esta activo indica que se esta estableciendo o procesando una conexión TCP. Si se combina con el bit ACK proporciona los segmentos específicos de control TCP.

1. $ACK=1 + SYN=1 \Leftrightarrow$ Se acepta la solicitud del establecimiento de la conexión.

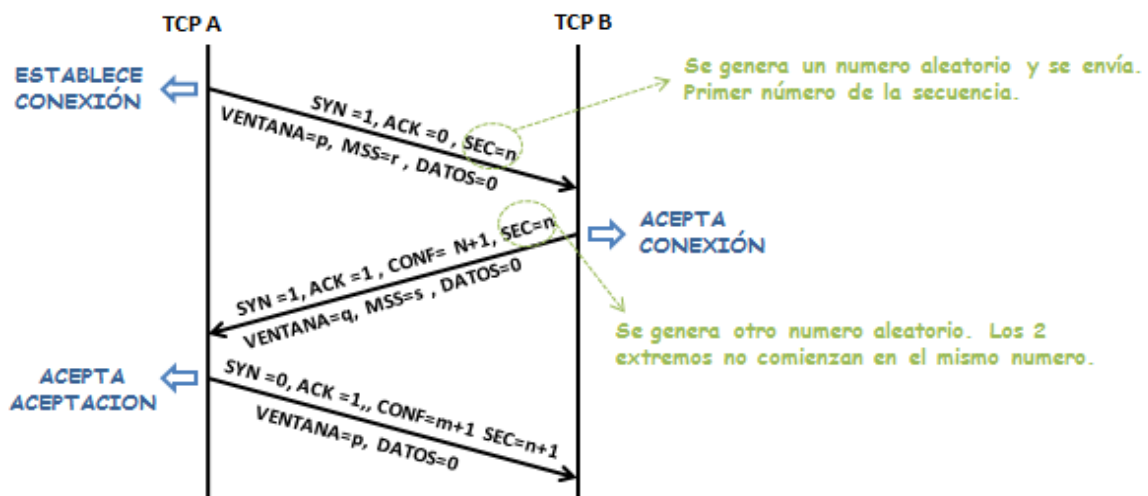
2. $ACK=1 + SYN=0 \Leftrightarrow$ Solicitud de establecimiento de conexión TCP.

* **VENTANA**: Se utiliza como control de flujo. Indica el numero de octetos pendientes de confirmación que el receptor puede manejar en función del tamaño puntual de su buffer de recepción, es decir, el numero de bytes que puede transmitir sin necesidad de esperar confirmación.

El tamaño máximo de la ventana será $Tam_Max = 2^{16}-1$.

• ESTABLECIMIENTO DE CONEXION:

Para establecer una conexión, se necesitan enviar tres segmentos de control en total.

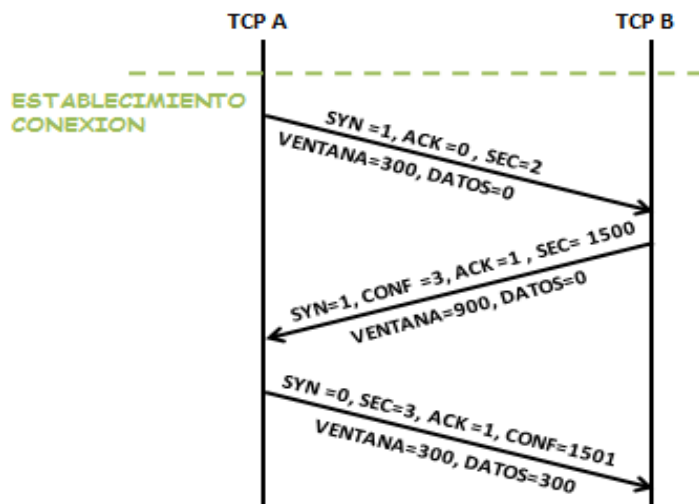


Los números de secuencia no se reutilizan durante un tiempo prudencial para evitar que los octetos de datos de los segmentos de información de una conexión se confundan con los de otra, sobre todo cuando se cierran y abren conexiones inmediatamente.

Los números de secuencia son aleatorios porque si no podrían ocurrir que segmentos obsoletos no se distingan de los actuales ya que los segmentos se pueden perder, retrasarse o duplicarse.

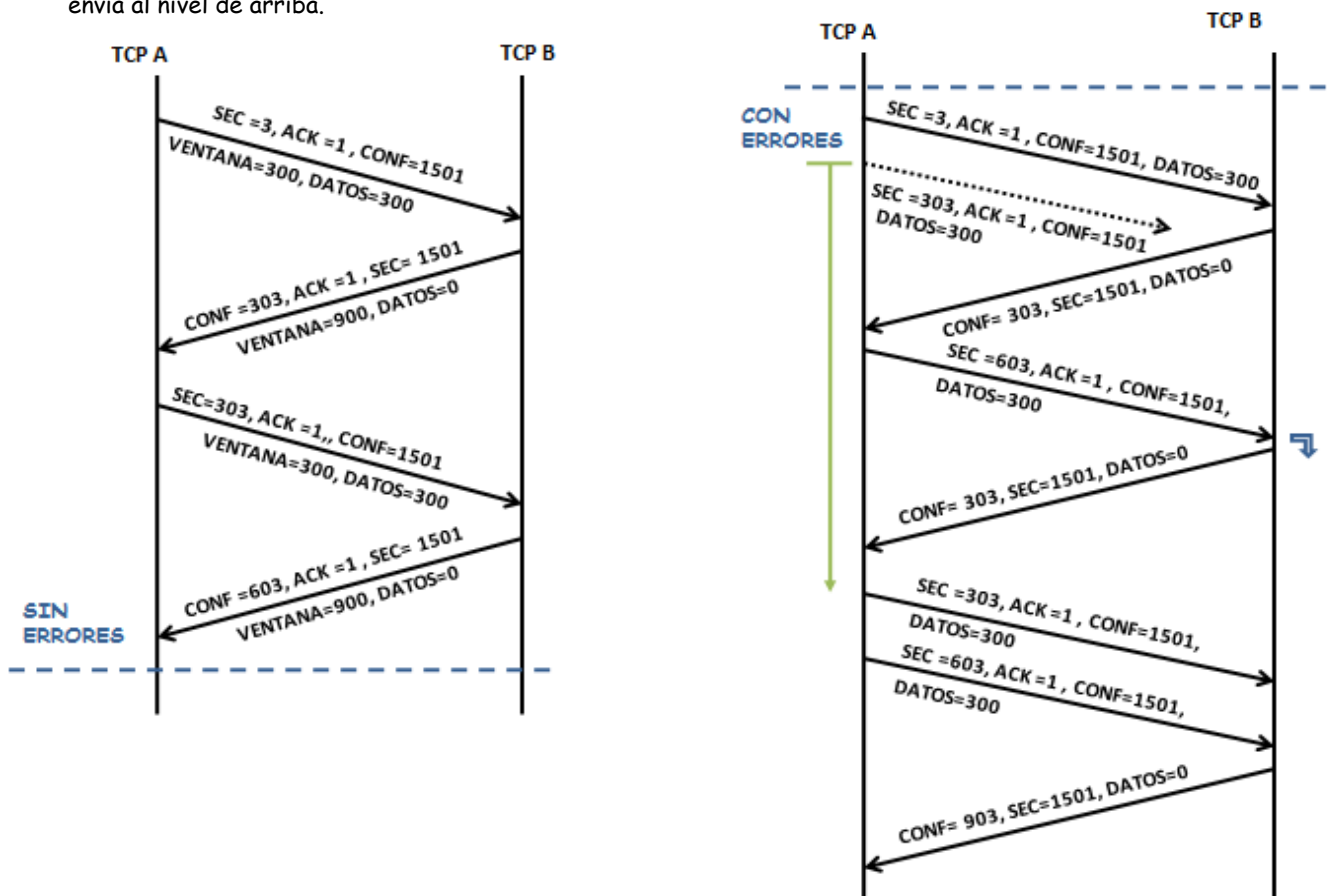
Se va a establecer el tamaño máximo de la ventana de recepción y opcionalmente el tamaño del campo de datos (MSS) de los segmentos que se esperan recibir.

* EJEMPLO:



• TRANSFERENCIA DE DATOS:

El intercambio de octetos entre aplicaciones se realiza mediante unos buffers de envío y recepción. Estos buffers permiten el almacenaje de los datos enviados o transmitidos. Hasta que el buffer de recepción no esté lleno no se envía al nivel de arriba.



En este caso nos vamos a basar en una transmisión unidireccional de datos independientemente que durante la transmisión se produzcan errores o no.

La ventana de recepción en A es de 300 octetos y en B de 900.

-SIN ERRORES:

1. Se envía un segmento desde A hacia B de información que contiene 300 octetos en el campo de datos (DATOS = 300) además recuerda a la entidad TCP B el tamaño de la ventana de recepción de A (300 octetos).
2. La entidad TCP B envía un segmento sin datos (DATOS = 0) en el que se confirman todos los octetos recibidos (CONF = 303)

El campo CONF contiene el siguiente número de octeto que se espera recibir, por tanto A entiende que todos los octetos enviados anteriormente han llegado bien. Además se recuerda el número de secuencia que va a utilizar y el tamaño de su ventana (900 octetos).

Estos dos pasos se van a producir con cada segmento de datos que se transmitan.

-CON ERRORES:

Suponemos que la entidad TCP A va a transmitir 900 octetos, agrupados en tres segmentos pendientes de confirmación.

Además vamos a suponer que el segundo segmento se va a perder por el camino, al no haber retransmisión se va a tener que esperar el vencimiento del temporizador.

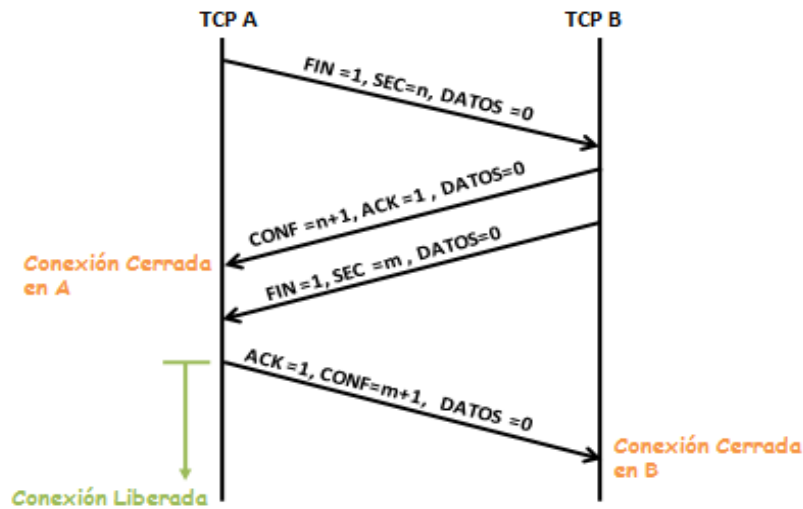
El tercer se va a enviar y recibir correctamente, pero como ha llegado fuera de secuencia se va a descartar.

Desde el momento que se pierde un segmento la entidad B va a estar enviando su segmento solicitando la transmisión del segmento perdido.

Una vez que vence el temporizador se van a retransmitir los segmentos dos y tres y la estación B va a confirmar únicamente el último (CONF = 903), esto hace que también se confirme el segmento dos.

- **LIBERACION DE CONEXION:** Se realiza de forma ordenada.

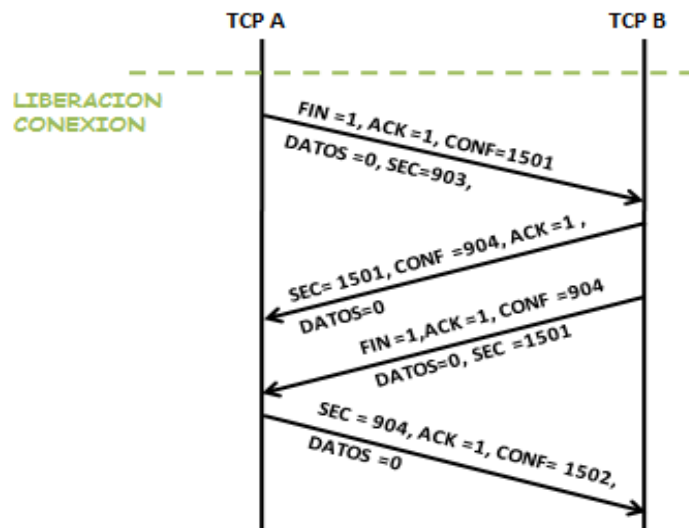
Para liberar cada lado de la conexión, es necesario haber recibido previamente una confirmación de todos los octetos enviados. Esta liberación ordenada implica un cierre independiente en cada dirección de la conexión y cualquiera de las partes puede solicitar la liberación de la conexión.



Las dos partes pueden iniciar la liberación simultáneamente, en este caso, la liberación se completa cuando una de las partes ha enviado una confirmación. Como no hay confirmaciones de las confirmaciones, al transmitirse la última confirmación se lanza un temporizador con un tiempo estimado de llegada de esa confirmación.

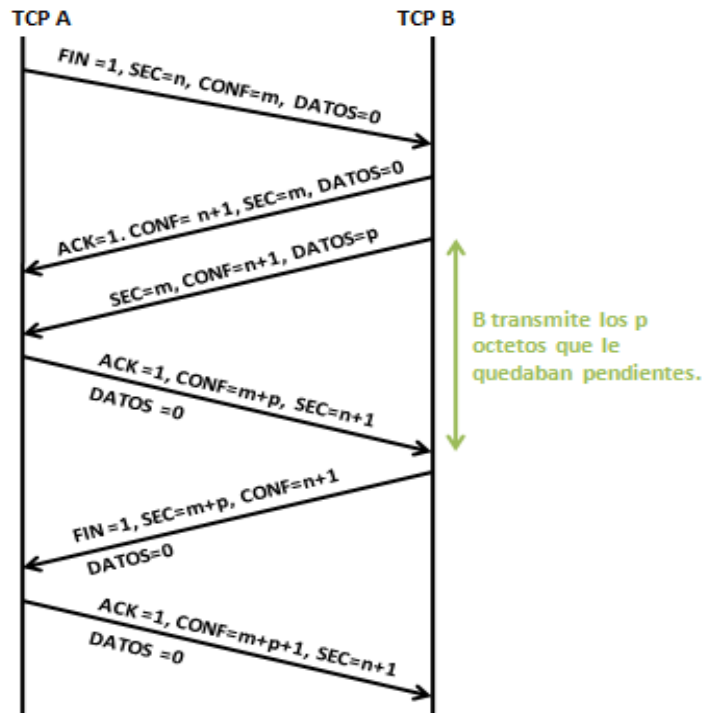
Cuando vence el temporizador se va a liberar oficialmente la conexión, este temporizador permite que dar un tiempo para que llegue la confirmación a su destino y a recibir potenciales segmentos retrasados u obsoletos para su eliminación inmediata.

* **EJEMPLO:** En este ejemplo vamos a considerar que no quedan datos pendientes por transmitir en ninguna de las dos estaciones.



La conexión se libera completamente cuando se transmite en cada sentido un segmento con el bit FIN activado.

-**LIBERACION DE CONEXIÓN CON DATOS:** Esta situación se da cuando una de las estaciones cierra la conexión por que ha terminado de transmitir datos y la otra continuar abierta hasta que termine de transmitir los datos que le quedan por enviar.



La estación B no va a cerrar la conexión hasta que primero no haya enviado todos los datos que tenga pendiente. Una vez que ya no quedan mas datos enviara una solicitud de liberación (FIN =1).

4.2 PROTOCOLO UDP.

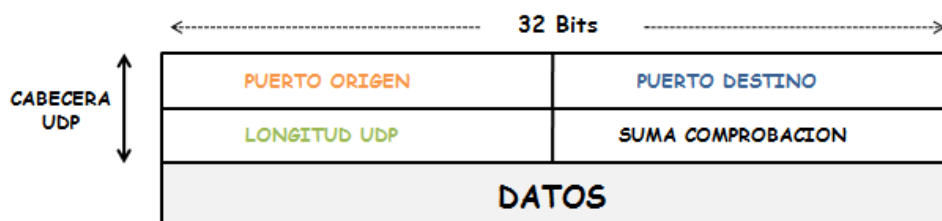
• CARACTERISTICAS:

1. No ofrece fiabilidad extremo a extremo.
2. Ofrece un servicio no orientado a conexión. No ofrece control sobre errores lógicos (detección y recuperación), no control de flujo.
3. Ofrece detección opcional, sin recuperación, de errores físicos. Opcionalmente se comprueba la integridad de la cabecera y datos del datagrama UDP, utilizando un método similar al de suma comprobación.
4. Multiplexación (Origen) y Demultiplexación (destino) en función de los números de puerto.
5. Transferencias en modo full_duplex.

• DATAGRAMA UDP:

-**DATOS:** Incluye la cabecera de información del nivel de aplicación y los datos del mensaje (si existen). Los datos deben pasarse en bloques bien delimitados ya que no existe un servicio de flujo de octetos como en TCP.

-**CABECERA:** Tiene longitud mínima de 8 octetos.



- * **PUERTO ORIGEN/DESTINO:** Limitados a 16 bits cada uno. Identifican al proceso de aplicación emisor / receptor que envía un segmento UDP.
- * **LONGITUD UDP:** Indica la longitud en octetos del datagrama UDP completo (cabecera y datos).
- * **SUMA COMPROBACION:** Suma aritmética binaria o en modulo 2 de todos los bloques de 16 bits del datagrama. Si una entidad UDP no desea calcular la suma el campo debe ser cero. Cuando en un datagrama se detecta un error se elimina y no se entrega al proceso de aplicación.